Standardi u proizvodnji softvera

U okviru ove, ali i narednih lekcija bavićemo se teorijom i standardima koji postoje kao osnova za organizaciju razvoja softvera, kao i upravljanja njegovim kvalitetom. U ovoj lekciji fokusiraćemo se na standarde koji se koriste prilikom planiranja i izrade softvera; dobićete uvid u to zašto su oni nastali i kako su nam korisni u svakodnevnom radu, a ujedno obavezni u radu u okviru kompanija i značajni u saradnji sa klijentima.

Osiguravanje kvaliteta u razvoju programa

Sa razvojem programiranja kao oblasti i napredovanjem u domenu razvoja i kompleksnosti softverskih rešenja, došlo je do sve veće potrebe za proverom ispravnosti, kvaliteta i brzine rada proizvoda. Rast timova, česte greške, rad sa više lokacija, objedinjavanje fajlova projekata, te stalna optimizacija i rast programa samo su neki od faktora koji su doveli do rasta potrebe za standardima, održavanjem kvaliteta i testiranjem.

Prvi i najvažniji pojam koji nam je potreban za dalji rad je *softversko testiranje*. Softversko testiranje možemo definisati kao proces verifikacije računarskog sistema ili programa i odlučivanje da li proizvod ispunjava određene zahteve i daje odgovarajući rezultat. Možemo reći da je ishod rada pronalaženje grešaka u kodu (bugova) u proizvodu ili na projektu.

Možda vam se nakon ove definicije taj posao i ne čini toliko važnim, jer možemo reći da kod testiramo svaki put kad napišemo neku funkcionalnost naše aplikacije. Međutim, pogledajmo neke od posledica koje su problemi u kodu izazvali na velikim projektima ili proizvodima:

Mt. Gox, najveća berza bitcoina

Nakon hakerskog upada, u junu 2011. godine, berza Mt. Gox je izgubila preko pola milijarde američkih dolara u vrednosti bitcoina. Ovaj bezbednosi nedostatak je na kraju doveo do zatvaranja kompanije.

Svemirska sonda Mariner 1

Mariner 1, predviđen za misiju oko Venere, skrenuo je sa kursa prilikom lansiranja, 1962. godine. Inženjeri Nase, zabrinuti da će letelica pasti na tlo, pokrenuli su proces samouništenja. Kasnijom analizom utvrđeno je da je u okviru koda postojala crtica viška, koja je poremetila precizne parametre navođenja rakete. Dakle, jedan karakter je izazvao propast misije i gubitak od tadašnjih 18 miliona dolara.

Nest pametni termostati

Update softvera za Nest pametne termostate je pošao po zlu i greška u kodu updatea je izazvala ubrzanu potrošnju baterija, što je na kraju dovelo do prestanka rada termostata. Mušterije nisu mogle da zagrevaju svoje domove; ovaj problem je zahvatio 99,5% korisnika, što je trajno uticalo na prodaju ovog proizvoda i na zadovoljstvo korisnika.

Sotfversko testiranje je neophodno jer svi pravimo greške. Nekada su te greške beznačajne, ali nekada mogu biti i skupe i opasne. Stoga je potrebno sve pažljivo proveriti, jer stvari mogu krenuti po zlu.

Možda sada mislite – u redu, ali programer može pažljivo da proveri svoj kod i uveri se da ne postoje greške. Problem je što nekada greške nastaju iz loše pretpostavke. U tom slučaju, možemo prilikom provere ponoviti iste greške kao i kada smo napisali kod. Stoga je važno da neko drugi proveri kod, jer će možda lakše uočiti greške koje mi nismo mogli. Navedeno je dovelo da uvođenja standarda i metodologija u okviru programiranja gde članovi jedne radne zajednice strogo poštuju pravila, što omogućuje lakši raspored obaveza, bolju organizaciju rada i ujedno i postizanje željenog nivoa kvaliteta proizvoda.

Pre nego što zakoračimo dalje u ovu oblast, važno je napomenuti da izučavanje i definisanje standarda i metodologija generalno pripada oblasti softverkog inženjerstva. Softversko inženjerstvo kao oblast obuhvata sve aspekte razvoja softvera.

Slika 1.1. Ilustracija razvoja softvera[1]

Dakle, softversko inženjerstvo obuhvata sve od planiranja zahteva, preko arhitekture i konstrukcije softvera, do testiranja i obezbeđivanja kvaliteta. Kako je ovo velika oblast IT sektora koja uključuje desetine drugih profesija, u okviru ovog kursa fokusiraćemo se na delove krucijalne za razvojne faze i testiranje sa obezbeđenjem kvaliteta koji vama, kao budućim programerima mogu biti potrebne u svakodnevnom radu.

ISO/IEC standardi

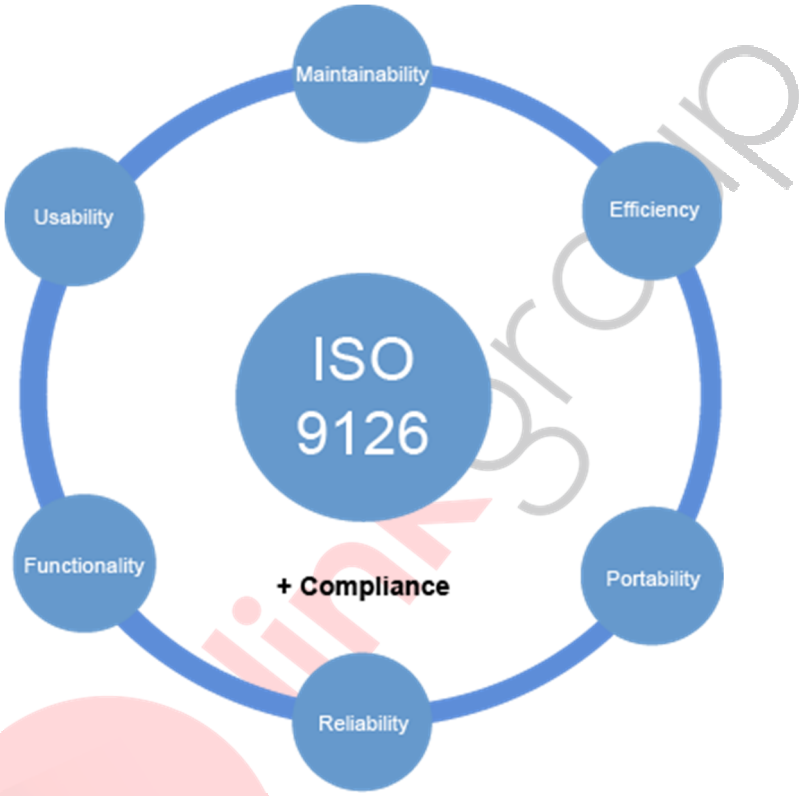
Pre svega, definišimo šta je to ISO i šta predstavlja ISO standard. ISO (International Organization for Standardization) jeste međunarodna organizacija za standardizaciju. Cilj ove organizacije je razvoj standarda za održavanje kvaliteta koji se primenjuju širom sveta. Pomoću ISO standarda, kompanija dobija precizno definisane parametre, ciljeve i zahteve koje je potrebno ispoštovati u nekom proizvodnom procesu ili prilikom pružanja usluge. Upravo iz ovoga razloga, mogli ste da vidite šifru nekog ISO standarda na npr. prehrambenim proizvodima, građevinskim mateirijalima, u bankama i slično.

Naravno, ISO obuhvata i druge aspekte, sve od zaštitne životne sredine, energetske efikasnosti, medicine do turizma. Nama najbitniji u ovom trenutku su ISO standardi koji se primenjuju u razvoju softvera i osiguravanju njegovog kvaliteta.

[1] https://www.freepik.com/free-vector/application-developmentbanner_1577664.htm#page=1&query=software%20engineering&position=7

ISO/IEC 9126 standard

ISO 9126 standard je nastao sa potrebom da se iz projekata razvoja softvera uklone česte greške koje nastaju usled ljudskih faktora, najčešće pogrešnih pretpostavki. Ove pretpostavke mogu biti vezane za prioritete u samom projektu, nedostatak tačne definicije šta se smatra uspešnim projektom i slično. Uloga standarda je da olakša definisanje prioriteta projekta, njegovih rokova i kranjih ciljeva koji moraju biti postignuti. Ciljevi mogu biti budžetski, vremenski, kao i ciljevi po pitanju funkcionalnosti softvera, tolerancije grešaka i postizanja željenog nivoa kvaliteta.



Slika 1.2. Kategorije ISO 9126 standarda

Na slici iznad možete videti glavne kategorije ISO standarda. One predstavljaju glavne faktore na koje se obraća posebna pažnja prilikom realizacije softvera u okviru ISO standarda. U okviru svake kategorije imamo određeni skup karakterisika, ali i poseban parametar – tzv. *compliance* parametar – kojim upoređujemo ishod sa planiranim ciljevima. U nastavku navodimo parametre svakog od faktora:

- Functionality – Kategorija funkcionalnosti predstavlja grupu parametara koji pokazuju da li program obavlja očekivanu funkciju. Ova grupa obuhvata parametre poput preciznosti, bezbednosti, prilagođenosti i kompatibilnosti, kao i compliance parametar, kojim utvrđujemo koliko realizovana funkcionalnost prati planiranu funkcionalnost programa.
- Reliability – Kategorija pouzdanosti obuhvata parametre kojima se određuje sposobnost programa da održi očekivane performanse rada. Ona obuhvata parametre poput tolerancije na greške i sposobnosti oporavka programa nakon greške, a naravno, kao i za svaku kategoriju, tu je i poseban parametar koji upoređuje ishod sa planiranim parametrima pouzdanosti.

- Usability – Faktor upotrebljivosti sadrži parametre kojima se određuje koliko je truda potrebno da korisnik uloži prilikom korišćenja programa; jednostavno rečeno – koliko je lako koristiti program. U okviru ove kategorije nalaze se parametri poput lakoće razumevanja, privlačnosti i jednostavnosti korisničkog interfejsa, kompleknosti korišćenja i, naravno, sada već poznatog compliance parametra.
- Maintainability – Ovim faktorom utvrđujemo koliko je truda potrebno da se izvrši promena ili održavanje programa. U okviru ovog faktora su parametri poput stabilnosti, lakoće analiziranja koda programa, modularnosti i prustupačnosti testiranju.
- Portability – Faktor prenosa određuje u kojem je stepenu moguće izvršiti prenos softvera sa jednog okruženja na drugo. Ovo je važan faktor za aplikacije za koje će biti očekivano da promene server ili uređaj na kojem se izvršavaju. Relevantni parametri su prilagodljivost, lakoća instalacije, uporedni rad sa drugim softverom sličnog tipa i, na samom kraju, faktor kojim upoređujemo krajnji rezultat sa ishodom.
- Efficiency – Faktor efikasnosti određuje vezu između performansi i količine resursa koje softver koristi. Među parametre ove kateogrije spadaju trajanje, iskoristivost resursa i, naravno, kao i kod prethodnih kategorija – compliance faktor, kojim upoređujemo krajnji rezultat sa ishodom.

Svaki od navedenih parametara se dalje deli na dodatne stavke u zavisnosti od softvera koji se kreira. Tada projektni menadžer, na samom početku projekta, u saradnji sa timom definiše precizne ciljeve koji se uređuju i definišu prema ISO standardu i nakon toga prate tokom celog procesa realizacije. Primera radi, incijalno će se dogovoriti šta tačno program treba da uradi, npr. da li podatke unosi korisnik ili administrator, koliko će brzo morati da se izvršava, da li će raditi na Android uređajima i na kojim verzijama sistema, kolika je tolerancija na greške i slično.

Sa godinama i napredovanjem razvoja softvera, razvijao se i ISO standard; od standarda ISO 9126, koji je godinama predstavljao osnovu standardizovanog razvoja softvera, došli smo do verzije 25010. Ova verzija predstavlja proširenje osnovnog standarda, sa jačim fokusom na stabilnost, kompatibilnost i bezbednost.

Iz ovog dela lekcije važno je razumeti koja je svrha jednog standarda, a to je – da nam pruži precizne parametre o kojima razmišljamo pre nego što započnemo pisanje koda softvera. Parametri standarda su tu da nam pomognu da se ne udaljimo od onoga što je cilj i da uporedimo ono što smo očekivali sa onim što smo dobili.

Da bi se ovakav posao organizovao između razvojnih timova, a ponekad i između par programera, tu su posebne metodologije koje se primenjuju tokom razvojnog procesa programa.

Metodologije razvoja softvera

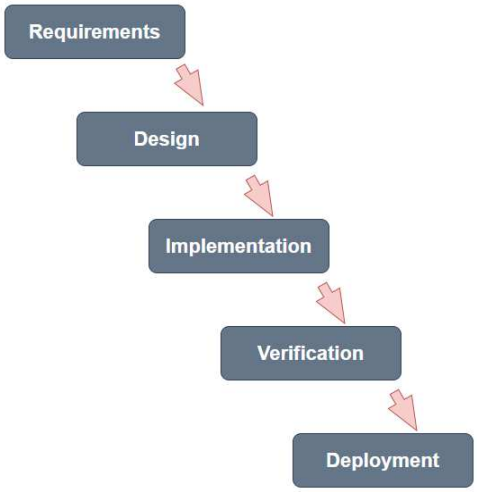
Krenućemo od pitanja – šta je to tačno metodologija razvoja softvera? Metodologiju možemo definisati kao skup procesa koji se koriste u razvoju softvera. Za razliku od standarda, kojima definišemo zahteve i ciljeve softvera, metodologijom definišemo kako će se rad odvijati.

Metodologija tačno definiše pravila kako se posao raspoređuje, ko će raditi da njemu i kako će zaposleni razmenjivati informacije u vezi sa projektom, bilo to dokumentacijom ili sastancima.

Možda ovo sada deluje apstraktno, ali pogledajmo jednu metodologiju koju smo do sada svi koristili, a da najverovatnije nismo ni znali da je to metodologija razvoja softvera.

Waterfall metodologija

Ova metodologija deli projekat na delove koji se kao aktivnosti obavljaju linearno u fazama, gde svaka faza zavisi od prethodne. Pogledajmo to na dijagramu:



Slika 1.3. Waterfall metodologija

Kao što možete videti, način realizacije projekta je vrlo jednostavan. Kada završimo jednu fazu, započinjemo drugu. Ovaj princip možemo primeniti i na svakodnevnicu, npr. na proces izrade domaćeg zadatka. Dobijamo zahteve zadatka u kojima je definisano šta je potrebno uraditi (requirements). Sledeća faza nam je da isplaniramo kako ćemo uraditi (design), a nakon toga krećemo sa implementcijom, dakle izradom samog zadataka. Kada to završimo, proveravamo (verification) šta je urađeno i da li je to traženo; ako jeste, poslednja faza nam je da predamo/pošaljemo svoj rad (deployment). Ovo je, naravno, uprošćeno radi lakšeg razumevanja, ali isti princip važi i za razvoj softvera. U tom slučaju dobijamo – ili sami precizno sastavljamo – zahteve šta softver treba da uradi; u dizajn fazi, brinemo o okruženju u kojem će softver raditi, korisničkom interfejsu i tehnologijama koje ćemo koristiti. U fazi implementacije, pišemo sam kod, dok u fazi verifikacije proveravamo da li program pravilno radi bez grešaka. Ako je sve ovo ispunjeno, softver šaljemo klijentu, odnosno postavljamo na server ili neku drugu lokaciju, što predstavlja deployment fazu.

Dakle, ovo je najjednostavnija metodologija razvoja softvera, koja se koristi već decenijama. Najveća mana ove metodologije je što je vrlo teško razmišljati o svim aspektima softvera na samom početku, dakle u reqiurements fazi, a ova metodologija ne predviđa da se vratimo na početak i unesemo promenu i samo nastavimo dalje – svaka promena će uticati na svaku od faza. Dakle, na promene u dokumentaciji, dizajn fazi i samoj implementaciji i verifikaciji.

Ukoliko bismo ovu metodologiju primenili u poslovnom okruženju – zamislite da imamo tim od deset programera, koji su utrošili šest meseci na realizaciju softvera, tek da bi se u fazi verifikacije utvrdilo da postoji neki nedostatak koji nije uočen tokom planiranja projekta. Ovo bi značilo da smo izgubili važan period za realizaciju i novac da bismo se vratili unazad i ponovo sve isplanirali.

Iz ovog razloga je došlo do potrebe da se za veće projekte i rad u timu koriste druge metodologije. Upravo te metodologije su sada standard u industriji. U nastavku ove lekcije, govorićemo o jednoj od njih, a u daljem toku kursa i o još par pristupa razvoju softvera.

Pitanje

ISO je kraći naziv za:

1. **International Organization for Standardization**
2. International Space Observatory
3. International Standards Organization

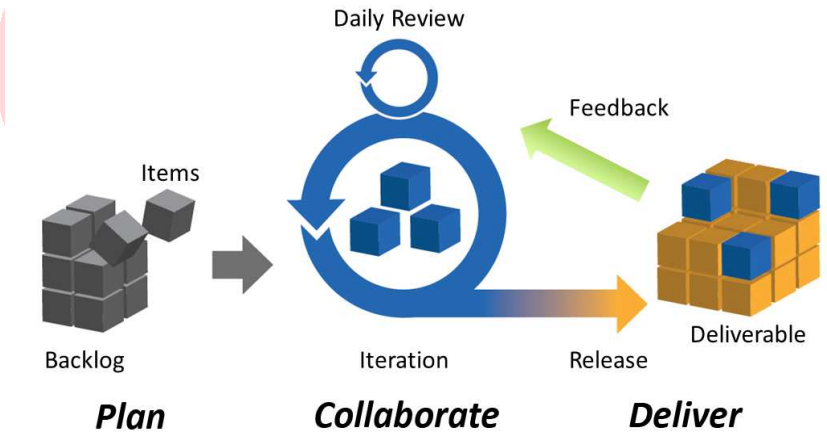
Objašnjenje:

ISO (International Organization for Standardization) jeste međunarodna organizacija za standardizaciju. Cilj ove organizacije je razvoj standarda za održavanje kvaliteta koji se primenjuju širom sveta. Pomoću ISO standarda, kompanija dobija precizno definisane parametre, ciljeve i zahteve koje je potrebno ispoštovati u nekom proizvodnom procesu ili prilikom pružanja usluge

Agile metodologija

Agile metodologija je jedan od predstavnika popularnih metodologija koje unapređuju proces izrade softvera. Timovi koriste agile metodologiju da bi se smanjio rizik od pojave grešaka (bugova), prelaska budžeta i promena u zahtevima softvera kada se dodaje nova funkcionalnost.

U okviru agile metodologije, timovi razvijaju softver u iteracijama od kojih svaka sadrži manja unapređenja funkcionalnosti i ove iteracije se ponavljaju sve dok se ne dobije završena funkcionalnost.



Slika 1.4. Ilustracija agilnog razvoja softvera[2]

[2] Author: Planbox, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=19543504

Agilni razvoj kreće od inicijalnog plana za softver, daleko manje detaljnog od npr. dokumentacije potrebe za waterfall metodologiju. Realizacija softvera se deli na male delove, od kojih više delova ulazi u proces realizacije. Za razliku od ostalih metodologija, u okviru agile metodologije, svaki pojedinačni deo proverava kako tester, tako i sam klijent. Na ovaj način se odmah dobija povratna informacija za te delove softvera. Ukoliko postoji problem sa nekim od delova, on se vraća u prvu fazu – backlog, gde se ponovo uvodi u realizaciju i dostavlja. Ovaj ciklus se ponavlja ukrug (iteracija) sve dok se svi delovi ne završe.

Ova metodologija omogućava da se softver dostavi u delovima. Iterativno dostavljanje delova koda poboljšava efikasnost tima kad je reč o pronalaženju i otklanjanju grešaka, jer se radi sa manjim delovima koda. Takođe, korisnici i klijent mogu bolje da prate proces kreiranja softvera i ostvarivanje ciljeva. Postoji manje planiranja i manje dokumentacije, što dovodi da kraćeg roka za realizaciju projekata.

Naravno, postoje i nedostaci primene ovakve metodologije. Prvenstveno, primena agilnih metoda razvoja otežava novim korisnicima korišćenje samog softvera. Naime, razvojni tim radi na delovima aplikacije i ne objavljuje dokumentaciju za završene delove. Usled toga, korisnici se sami snalaze u korišćenju i testiranju programa sve dok se ne dobije završen proizvod. Primena ovih metoda nad velikim projektima otežava procenu za koliko će softver biti završen, jer upravo nema planiranja između delova već se greške otklanjaju u toku rada.

I pored navedenih nedostataka, agile metodologija je danas jedan od najpopularnijih principa razvoja softvera. Iz ove metodologije su proizišle mnoge metode rada, uključujući scrum, extreme programming, feature-driven development i test-driven development, o kojima ćemo govoriti u nastavku ovog kursa.

Dakle, u ovoj lekciji smo govorili o tome šta je to standard u razvoju softvera, te kako se metodologija povezuje sa standardom. Bilo je reči o najopopularnijim metodologijama i njihovim prednostima i manama. Sve ovo je uvod za lekcije koje slede, gde ćemo detaljnije govoriti o različitim aspektima testiranja i kvaliteta.

Rezime

- Softversko inženjerstvo obuhvata sve aspekte razvoja softvera – od planiranja zahteva, preko arhitekture i konstrukcije softvera, do testiranja i obezbeđivanja kvaliteta.
- Softversko testiranje kao oblast softverskog inženjerstva možemo definisati kao proces verifikacije računarskog sistema ili programa i odlučivanje da li proizvod ispunjava određene zahteve i daje odgovarajući rezultat.
- ISO (International Organization for Standardization) jeste međunarodna organizacija za standardizaciju. Cilj ove organizacije je razvoj standarda za održavanje kvaliteta koji se primenjuju širom sveta. Pomoću ISO standarda, kompanija dobija precizno definisane parametre, ciljeve i zahteve koje je potrebno ispoštovati u nekom proizvodnom procesu ili prilikom pružanja usluge.
- ISO 9126 standard je nastao sa potrebom da se iz projekata razvoja softvera uklone česte greške koje nastaju usled ljudskih faktora, najčešće pogrešnih pretpostavki.
- ISO 9126 standard se sastoji iz šest osnovih kategorija: Functionality, Reliability, Usability, Maintainability, Portability i Efficiency. Svaka od ovih kategorija sadrži parametre koji se dodatno dele u posebne stavke u zavisnosti od softvera koji se kreira.
- Kada pratimo određeni standard kvaliteta, postoji potreba za posebnom organizacijom posla, te koristimo specifične metodologije razvojnog procesa. Metodologijom

definišemo kada će se rad odvijati, kako se posao raspoređuje, ko će raditi da njemu i kako će zaposleni razmenjivati informacije u vezi sa projektom.

- Waterfall metodologija je jedna od najjednostavnijih metodologija razvoja; decenijama je prisutna u razvoju softvera. U okviru nje, projekat se deli na delove koji se kao aktivnosti obavljaju linearno u fazama. Kada završimo jednu fazu, započinjemo drugu.
- Agile metodologija je jedan od predstavnika popularnih metodologija. Timovi koriste agile metodologiju da bi se smanjio rizik od pojave grešaka (bugova), prelaska budžeta i promena u zahtevima softvera kada se dodaje nova funkcionalnost.
- Agilna metodologije se zasniva na razvoju softvera u iteracijama, od kojih svaka sadrži manja unapređenja funkcionalnosti; ove iteracije se ponavljaju sve dok se ne dobije završena funkcionalnost.